Интегрисани информациони систем за издавање диплома и конкурс за упис на високошколске установе еФакултет

Борисав Живановић

4. јул 2023.

Сажетак

У раду је описан интегрисани информациони систем за издавање диплома и конкурс за упис на високошколске установе еФакултет. Применом овог решења омогућено је централизовано издавње и праћење издатих диплома акредитованих вискошколских установа. Конкурс за упис се ослања на централизоване регистре диплома средњих школа и високошколских установа.

Кључне речи

информациони систем, високо образовање, факултет, диплома, еУправа

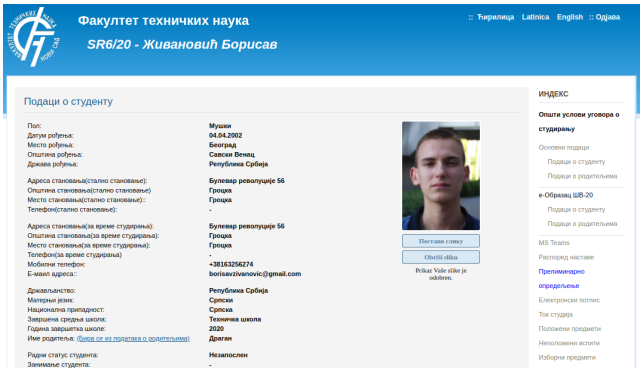
Увод

Конкурисање за упис на високошколске установе захтева подношење дипломе претходног нивоа образовања. Такође, конкурисање за посао често захтева доказ о завршеном степену школовања за одређену професију. Традиционални приступ подношења папирних диплома је подложан злоупотреби, како због подношења фалсификованих диплома, тако и због немогућности праћења издатих диплома. Аутоматизовано и централизовано праћење података о издатим дипломама омогућава већу транспарентност у односу на традиционални приступ и олакшава праћење издатих диплома.

Сродна истраживања

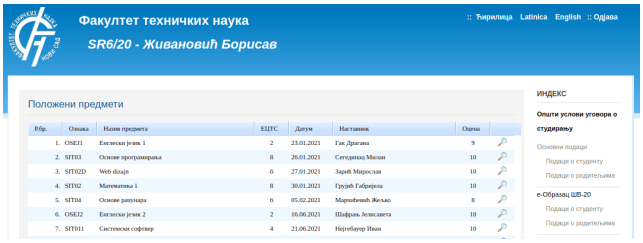
Већина факултета има информациони систем студентске службе у којем се чувају подаци о издатим дипломама матичног факултета. На је Figure 1, Figure 2, Figure 3 дат приказ изгледа информационог система студентске службе Факултета техичких наука Универзитета у Новом Саду. Мањи број факултета (попут Факултета техичких наука Универзитета у Новом Саду) имају информациони систем за конкурс, али као доказ о

завршеном претходном нивоу школовања се подносе скениране папирне дипломе. Такав приступ захтева ручну проверу исправности диплома и повећава вероватноћу за грешком (погрешно унет просек, прихватање фалсификоване дипломе).



Слика 1: Информације о студенту (студентска служба ФТН УНС)

Слика 2: Информације о току студија (студентска служба ФТН УНС)



Слика 3: Положени испити (студентска служба ФТН УНС)

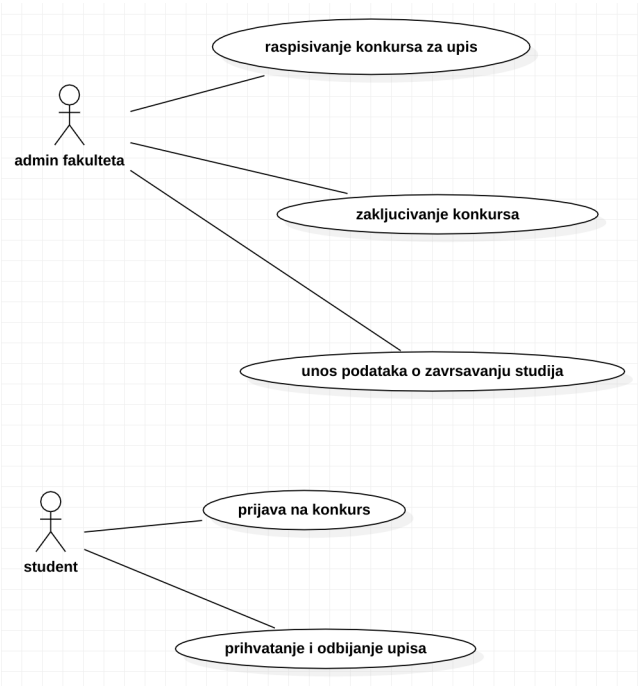
Коришћене технологије

За имплементацију клијентске апликације коришћен је програмски језик TypeScript[6] и окружење (енгл. framework) Angular[5]. За имплементацију клијентске апликације коришћен је програмски језик Java[7] и окружење Spring[9], уз библиотеку за објектно-релационо мапирање Hibernate[1]. За ауторизацију је коришћен централизовани SSO

(Single Sign On)[1] сервер еУправе (модификована верзија протокола OAuth 2.0[2]). За складиштење података коришћена је MySQL[8] база података.

Спецификација захтева

Корисници система су администратор и студент. Администратор расписује конкурс (са одговарајућим квотама) и окончава пријаве, а потом и конкурс. По окончавању конкурса, систем рангира пријаве по просеку, и пријављени из горњег дела ранг листе (по квоти) добијају могућност за упис. Администратор уноси податке о завршетку студија. Студент се пријављује по конкурсу, уз диплому из регистра. По изласку резултата, прихвата или одбија упис (уколико је распоређен по квоти за упис). Слика Figure 4 представља дијаграм случајева коришћења. Случајеви коришћења су детаљније описани у табелама испод.



Слика 4: Дијаграм случајева коришћења

[1]SSO омогућава употребу истих креденцијала за пријаву на више сервиса. Креденцијали се складиште на SSO серверу. Представља концепт, а не стандард који диктира начин имплементације. За више детаља погледати: How Does Single Sign-On Work?

Случај коришћења
Расписивање конкурса за упис
Типови корисника
администратор
Предуслови
1. Администратор је пријављен
Кораци
1. За сваки смер се уноси квота 2. Потврђује се расписивање конкурса
Последице
1. Конкурс је расписан

Случај коришћења
Закључивање конкурса
Типови корисника
администратор
Предуслови
1. Конкурс је расписан
Кораци
1. Администратор потврђује крај конкурса
Последице
1. Конкурс је затворен 2. Студенти су распоређени

Случај коришћења
Унос података о завршавању студија
Типови корисника
администратор
Предуслови
1. Студент је уписан на смер
Кораци
1. Администратор уноси податке о просеку
Последице
1. Студенту се приписује нови ниво образовања

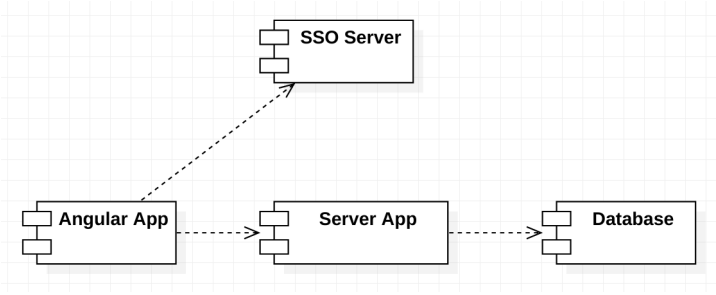
Случај коришћења
Пријава на конкурс
Типови корисника
студент
Предуслови
1. Студент је завршио потребан претходни ниво школовања
Кораци
1. Студент бира смер
Последице
1. Студент је пријављен на конкурс

Случај коришћења
Прихватање и одбијање уписа
Типови корисника
студент
Предуслови
1. Администратор је окончао конкурс 2. Студент је распоређен на смер
Кораци
1. Студент прихвата/одбија упис
Последице
1. Студент је уписан на смер

Спецификација дизајна

Архитектура система

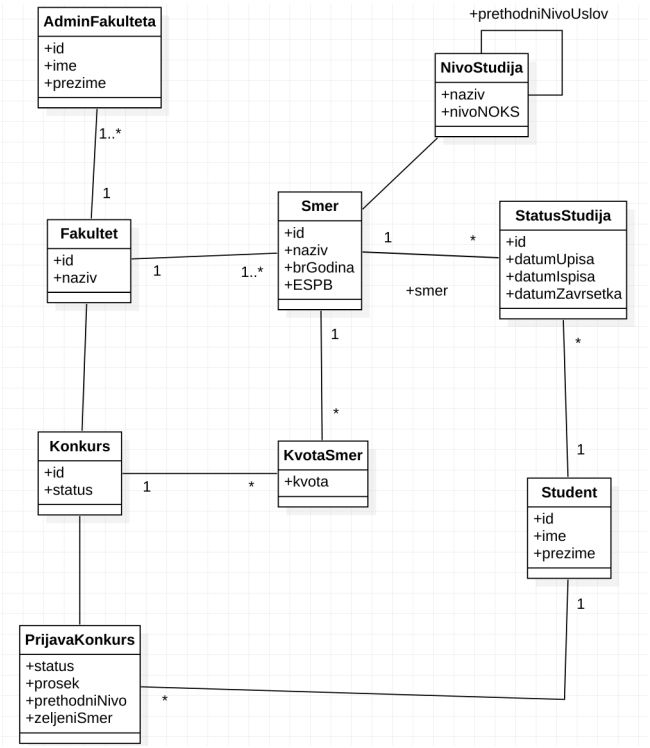
Систем је реализован као веб апликација. Клијентска апликација комуницира преко HTTP API-ја са серверском апликацијом. API је дизајниран у складу са REST архитектонским стилом[4]. Слика Figure 5 приказује архитектуру система.



Слика 5: Архитектура система

Модел података

На слици Figure 6 је приказан модел података иницијалне имплементације система. Модел података задовољава трећу нормалну форму, са изузетком пријава на конкурс, где се у оквиру пријаве чува просек са претходног нивоа образовања.



Слика 6: Модел података

Класама Admin и Student представљени су корисници система, а њиховим везама са осталим ентитетима одређена су њихова права приступа. У систему се не чувају лозинке.

Класа PrijavaKonkurs представља учествовање студента на конкурсу кроз све фазе (расписан, затворене пријаве, затворен конкурс). По затварању конкурса, систем рангира пријаве по просеку и прихваћеним пријавама поставља статус PRIHVACEN.

Класа Studiranje представља период студирања одређеног смера (представљеног класом Smer). Уколико је ниво студија успешно окончан, садржи датум завршетка и просек. Представља диплому којом је могуће учествовати у конкурсу за упис на наредни ниво студија.

Класа Konkurs представља један конкурс за упис на факултет. Квота 0 значи да се у датом конкурсу не врши упис студената за дати смер.

Имплементација

Клијент

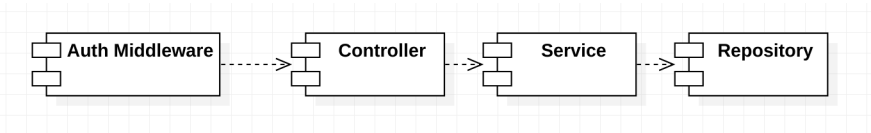
Клијентска апликација садржи администраторске и студентске функционалности, којима је приступ могућ у зависности од типа корисника. Тип корисника и привилегије су садржане у токену за ауторизацију. Приступ страницама је контролисан уз guard механизам који прижа Angular.

По потреби, опције су приказане или сакривене као у примеру.

Сервер

Серверска апликација омогућава приступ систему кроз REST API[4]. За комуникацију са сервером, неопходан је валидан токен издат од стране еУправа SSO сервера.

Компоненте серверске апликације су приказане на слици Figure 7.



Слика 7: Архитектура серверске апликације

Компонента AuthMiddleware преузима токен из заглавља Authorization из захтева и проверава исправност. Уколико је токен исправан, захтев се прослеђује даље у систем. Уколико токен није исправан, захтев се одбија уз грешку 403 Forbidden.

Компонента Controller (Listing 1) прослеђује захтев са одговарајућег REST API[4] на одговарајућу методу сервисног слоја. Садржај захтева преводи у одговарајуће структуре података програмског језика Java и резултат позива преводи у одговарајући HTTP одговор. Садржи грубу логику за ауторизацију по RBAC[3] моделу. Те провере нису довољне, али смањују оптерећење базе података и олакшавају читљивост кода.

```
@GetMapping("/{id}")
public Fakultet getFakultet(@PathVariable long id) {
    return fakultetService.getFakultet(id);
}
```

Listing 1: Пример Controller методе

Компонента Service (Listing 2) садржи апликативну логику. Једна сервисна метода одговара једној корисничкој акцији. Садржи комплетне провере права приступа на нивоу објекта, уз ослонац на ABAC[10] модел и уз додатне провере апликативне логике (пример: нису могуће две пријаве истог студента на исти конкурс). У тренутној имплементацији,

дата је предност употреби објектно-релационог мапирања у односу на писање SQL упита због разумљивости кода и брзине имплементације.

```
public long raspisiKonkurs(long fakultetId, Konkurs konkurs) {
    Fakultet fakultet = fakultetRepository.getById(fakultetId);

    konkurs.setFakultet(fakultet);
    konkurs.setDatumRaspisivanja(LocalDate.now());

    for(KvotaSmer kvotaSmer : konkurs.getKvote()) {
        kvotaSmerRepository.save(kvotaSmer);
    }

    konkursRepository.save(konkurs);

    return konkurs.getId();
}
```

Listing 2: Пример Service методе

Компонента Repository (Listing 3) садржи конкретне упите ка бази података. Једна метода одговара једном упиту. Дата је предност употреби JPQL у односу на SQL због веће читљивости кода и једноставнијег преласка на други систем за управљање базом података (енгл. Database Management System). Употребом парамтеризованих упита је спречена могућност извођења SQL injection напада [2].

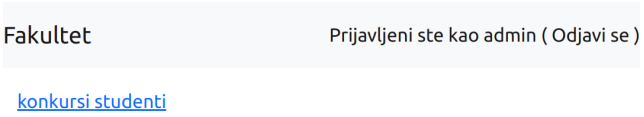
```
@Query("SELECT k FROM Konkurs k WHERE k.fakultet.id = ?1")
List<Konkurs> getKonkursiByFakultetId(long fakultetId);
```

Listing 3: Пример Repository методе

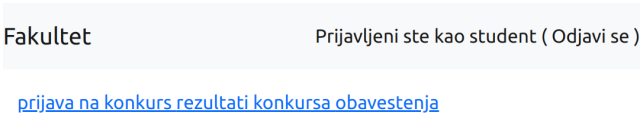
Демонстрација

По пријављивању у систем, кориснику је приказан панел који одговара његовој улози (Figure 8, Figure 9). Администратор има приказ конкурса његове установе (Figure 10), као и могућност расписивања нових конкурса (Figure 11). Студент има приказ свих тренутно активник конкурса са свих факултета (Figure 12), као и могућност пријаве на конкурс.

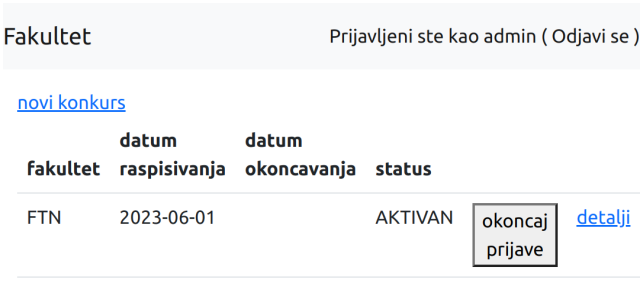
[2] SQL injection представља напад у којем нападач злоупотребљава наивни механизам конкатенације стрингова у формирању упита ка бази података. Иако апликација може да садржи адекватну логику за ауторизацију, њено заобилажење је могуће уколико дозволимо нападачу да утиче на креирање упита. За више детаља погледати: SQL injection



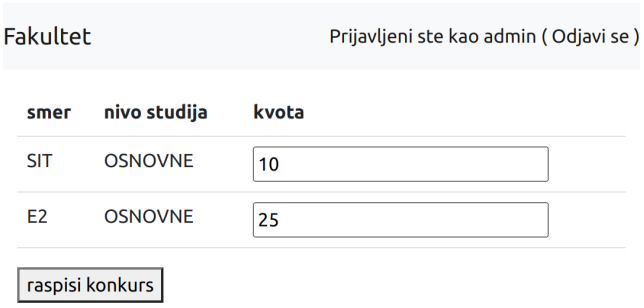
Слика 8: Администраторски панел



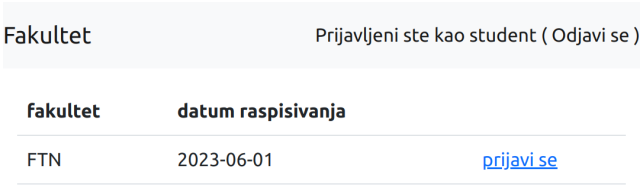
Слика 9: Студентски панел



Слика 10: Администраторска страница са конкурсима



Слика 11: Расписивање конкурса



Слика 12: Приказ свих активних конкурса

Закључак

У раду је описан интегрисани информациони систем за издавање диплома и конкурс за упис на високошколске установе еФакултет. Приказано је како је применом централизованих регистара диплома средњих школа и високошколских установа могуће убразати конкурс, као и смањити могућност грешки и представљања фалсификованих диплома. У тренутној имплементацији, регистар диплома и конкурс су обједињени у један систем. У будућности, могуће је раздвајање на две групе система: централизовани државни регистар диплома и на појединачне системе за конкурс (прилагођене специфичним потребама и правилима конкурса). Могућ правац развоја би био омогућавање јавног приступа подацима о завршеним степенима образовања државних функционера, службеника као и осталих лица од важности (лекара, наставника). За имплементацију такве функционалности би било потребно доношење додатних законских оквира.

Литература

[1] Hibernate, 2023. Приступљено 25.5.2023. URL: `https://hibernate.org/`.

[2] Ed. D. Hardt. The OAuth 2.0 Authorization Framework, 2012. Приступљено 31.5.2023. URL: `https://www.rfc-editor.org/rfc/rfc6749`.

[3] D. Richard Kuh David F. Ferraiolo. Role-Based Access Controls, 1992. Приступљено 31.5.2023. URL: `https://www.researchgate.net/publication/24164143_Role-Based_Access_Controls`.

[4] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures, 2000. Приступљено 31.5.2023. URL: `https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

[5] Google. Angular, 2023. Приступљено 25.5.2023. URL: `https://angular.io/`.

[6] Microsoft. TypeScript, 2023. Приступљено 25.5.2023. URL: `https://www.typescriptlang.org/`.

[7] Oracle. Java, 2023. Приступљено 25.5.2023. URL: https://www.java.com/.

[8] Oracle. MySQL, 2023. Приступљено 25.5.2023. URL: https://www.mysql.com/.

[9] Spring. Spring Framework, 2023. Приступљено 25.5.2023. URL: https://spring.io/.

[10] David F. Ferraiolo Vincent C. Hu, D. Richard Kuhn. Attribute-Based Access Control, 2015. Приступљено 31.5.2023. URL: https://www.researchgate.net/publication/273393378_Attribute-Based_Access_Control.